AMENDMENTS TO THE CLAIMS

Please amend the claims as follows:

1. – 39. (Canceled)

40. (Currently Amended) A computer program product encoded in at least one computer readable medium, the computer program product comprising:

at least one function sequence implementing an access operation on a concurrent shared ~~object~~ double-ended queue (deque), the ~~concurrent shared object~~ deque ~~instantiable~~ implemented as a circular buffer ~~of~~ on a contiguous array of bounded size ~~implementing a contiguous array~~ delimited by a pair of end identifying indices, wherein each of the pair of end identifying indices identifies an element adjacent to one of two end elements of the deque;

instances of the at least one function~~al~~ sequence concurrently executable by plural processors of a multiprocessor and each including an atomic dual target compare and swap (DCAS) to update a corresponding one of the pair of end identifying indices and an element of the array corresponding to a then-current value thereof; and

the DCAS of the at least one functional sequence responsive to a corresponding boundary condition state of the ~~concurrent shared object~~ deque.

41. (Currently Amended) [[A]] The computer program product as recited in 40,

wherein the at least one functional sequence ~~includes~~ is at least one selected from a group consisting of ~~opposing end variants of push and pop operations on the concurrent shared object~~ a right pop operation, a left pop operation, a right push operation, and a left push operation;

wherein the boundary condition state corresponding to the right push operation and the left push operation[[s]] is a full state of the ~~array~~ deque; and

wherein the boundary condition state corresponding to the right pop operation and the left pop operation[[s]] is an empty state of the ~~array~~ deque.

42. (Cancelled)

43. (Currently Amended) An apparatus comprising:

plural processors;

a store addressable by each of the plural processors;

first- and second-end index stores accessible to each of the plural processors for identifying opposing ends of a ~~bounded-size contiguous array~~ double-ended queue (deque) encoded in circular buffer form on a bounded-size contiguous array in the addressable store, wherein the first- and second-end index stores identify elements adjacent to the ends of the deque; and

means for coordinating competing access operations, the coordinating means employing in each instance thereof, at least one atomic dual target compare and swap (DCAS) operation to disambiguate a retry state and a boundary condition state of the ~~array~~ deque based on then-current contents of one, but not both, of first- and second-end index stores and an array element corresponding thereto.

44. (New) The apparatus of claim 43, wherein the access operations are selected from a group consisting of a right pop operation, a left pop operation, a right push operation, and a left push operation.

45. (New) The apparatus of claim 43, wherein each of the first- and second-end index stores identifies a next element in the array for adding a value to the deque.

46. (New) The computer program product of claim 40, wherein each of the pair of end-identifying indices identifies a next element in the array for adding a value to the deque.

47. (New) A method of managing concurrent access to shared data, comprising:

implementing a concurrent shared double-ended queue (deque) as a contiguous bounded-size array in a memory of a computer system, wherein a first end identifying index identifies a first array element adjacent to an end element of the deque and a second end identifying index identifies a second array element adjacent to the end element of the deque; and

performing a first pop operation on the deque, wherein a first atomic dual target compare and swap (DCAS) is executed using the first end identifying index and the end element, wherein the end element is removed from the deque.

48. (New) The method of claim 47, further comprising;

performing a second pop operation on the deque concurrently with the first pop operation, wherein a second DCAS is executed using the second end identifying index and the end element, wherein when the DCAS fails, an indication of an empty state of the deque is returned from the DCAS.

49. (New) The method of claim 47, wherein the deque is implemented as a circular buffer.

50. (New) The method of claim 48, wherein the first pop operation is performed by a first processor and the second pop operation is performed by a second processor.

51. (New) The method of claim 48, wherein the first pop operation is a left pop operation, the first end identifying index is a left-end index, and the first array element is to the left of the end element, and wherein the second pop operation is a right pop operation, the second end identifying index is a right-end index, and the second array element is to the right of the end element.

52. (New) A method of managing concurrent access to shared data, comprising:

implementing a concurrent shared double-ended queue (deque) as a contiguous bounded-size array in a memory of a computer system, wherein the deque comprises a first end element, a second end element, a first end identifying index, and a second end identifying index, and wherein the first end identifying index identifies an array element as adjacent to the first end element and the second end identifying index identifies the array element as adjacent to the second end element; and

performing a first push operation on the deque, wherein a first atomic dual compare and swap (DCAS) is executed using the first end identifying index, the array element, and a value, wherein the value is stored in the array element.

53. (New) The method of claim 52, further comprising:

performing a second push operation on the deque concurrently with the first push operation, wherein a second DCAS is executed using the second end identifying index and the array element, wherein when the DCAS fails, an indication of a full state of the deque is returned from the DCAS.

54. (New) The method of claim 52, wherein the deque is implemented as a circular buffer.

55. (New) The method of claim 53, wherein the first push operation is performed by a first processor and the second push operation is performed by a second processor.

56. (New) The method of claim 53, wherein the first push operation is a left push operation, the first end identifying index is a left-end index, and the first end element is to the right of the array element, and wherein the second push operation is a right push operation, the second end identifying index is a right-end index, and the second end element is to the left of the array element..